

CHAPTER

8

Exploring the UNIX Utilities

case ► Your supervisor at Dominion Consulting asks you to accomplish several computer tasks, including creating a bootable floppy disk, making backups on floppy disks, auditing system performance, analyzing hard disk storage, and removing unnecessary files. Further, you have been asked to create a man page to describe the Phmenu program that you wrote in Chapter 7.

LESSON A

objectives

In this lesson you will:

- Define the classifications of the UNIX utility programs
- Duplicate a floppy disk
- Create a “bootable floppy” to boot the system in case of emergency
- Determine hard disk usage and available free space
- Locate and remove unnecessary files from the hard disk
- Display the CPU status and internal memory usage

Using the UNIX Utilities

In this chapter, you continue to explore UNIX utilities. Utilities are designed to provide basic services to UNIX users, just like your local utility companies provide basic services (e.g., electricity and running water) to your community. This chapter lets you practice using UNIX commands and utilities, such as using the man command with UNIX’s text formatting utilities. Before you can begin to carry out your assigned tasks for Dominion Consulting, you need to learn more about UNIX utilities.

Understanding UNIX Utilities

UNIX utilities let you create and manage files, run programs, produce reports, and generally interact with the system. Beyond these basics, the utility programs offer a full range of services that let you monitor and maintain the system and recover from a wide range of errors. UNIX utilities are classified into seven major function areas dictated by user needs: file processing, networking, communications, system status, programming, source code management, and miscellaneous. UNIX utilities are programs, but they are usually referred to as commands.

Note: For the sake of completeness, this chapter contains some references and commentary about all the utilities, but it concentrates on those utilities that relate to file processing, system status, and miscellaneous tasks.

Utility programs are distinguished from other operating system programs in that they are “add-ons” and not part of the UNIX shells or a component of the kernel. Utilities are also unique because they are exclusively dedicated to improving computer performance for the benefit of users. New utility programs are continually being added as developers find better and faster ways to make UNIX run more efficiently. Finally, most UNIX utilities are small programs that often consist of only one command.

Classifying UNIX Utility Programs

Utilities can be classified in several categories, as some work exclusively with UNIX files, others handle network tasks, and still others are designed to help programmers. File processing utilities, listed in Table 8-1, is the largest category. These utilities display and manipulate files.

Utility	Brief Description of Function
afio	Creates an archive or restores files from archive
awk	Processes files
cat	Displays or joins files
cmp	Compares two files
comm	Compares sorted files and shows differences
cp	Copies files
cpio	Copies and backs up files to an archive
cut	Selects characters or fields from input lines
dd	Copies and converts input records
diff	Compares two text files and shows differences
fdformat	Formats a floppy disk at a low level
find	Finds files within file tree
fmt	Formats text very simply
grep	Matches patterns in a file
groff	Processes embedded text formatting codes
gzip	Compresses or decompresses files
head	Displays the first part of a file
ispell	Checks one or more files for spelling errors
less	Displays text files (pauses when screen is full)
ln	Creates a link to a file
lpr	Prints file (hard copy)
ls	Lists file and directory names and attributes
man	Displays documentation for commands

Table 8-1: File processing utilities

Utility	Brief Description of Function
mkdir	Creates a new directory
mkfs	Builds a UNIX file system
mv	Renames and moves files and directories
od	Dumps formatted file
paste	Concatenates files horizontally
pr	Formats text files for printing and displays them
rdev	Queries or sets the root image device
rm	Removes files
rmdir	Removes directory
sed	Edits streams (non-interactive)
sort	Sorts or merges files
tail	Displays the last lines of files
tar	Copies and backs up files to a tape archive
touch	Changes file modification dates
uniq	Displays unique lines of sorted file
wc	Counts lines, words, and bytes

Table 8-1: File processing utilities (*continued*)

System status utilities, listed in Table 8-2, is the second largest category. It includes utilities that display and alter the status of files, disks, and the overall system. These utilities let you know who is online, the names and status of running processes, the amount of hard disk space available, and where to find other commands you need to run.

Utility	Brief Description of Function
chgrp	Changes the group associated with a file
chmod	Changes the access permissions of a file
chown	Changes the owner of a file
date	Sets and displays date and time

Table 8-2: System status utilities

Utility	Brief Description of Function
df	Displays the amount of free space remaining on disk
du	Summarizes file space usage
file	Determines file type (e.g., shell script, executable, ASCII text, and others)
finger	Displays detailed information about users who are logged on
free	Displays amount of free and used memory in the system
kill	Terminates a running process
ps	Displays process status by process identification number and name
sleep	Suspends execution for a specified time
top	Dynamically displays process status, like ps, but in real time
w	Displays detailed information about the users who are logged on
who	Displays brief information about the users who are logged on

Table 8-2: System status utilities (*continued*)

Network utilities, listed in Table 8-3, consist of the essential commands for communicating and sharing information on a network.

Utility	Brief Description of Function
ftp	Transfers files over a network
rcp	Remotely copies a file from network computer
rlogin	Logs on to a remote computer
rsh	Executes commands on a remote computer
rwho	Displays the names of users attached to a network
telnet	Connects to remote computer on a network

Table 8-3: Network utilities

The communication utilities, listed in Table 8-4, handle the mail and messaging tasks. These programs include some recent advanced features such as **Multipurpose Internet Mail Extensions (MIME)** support for sending and receiving binary files in mail messages.

Utility	Brief Description of Function
mail	Sends electronic mail messages
mesg	Denies (mesg n) or accepts (mesg y) messages
pine	Sends and receives electronic mail and news
talk	Lets users simultaneously type messages to each other
write	Sends a message to another user

Table 8-4: Communications utilities

Programming utilities, listed in Table 8-5, are designed to help users develop software projects written in C and C++ programs.

Utility	Brief Description of Function
configure	Configures program source code automatically
gcc	Compiles C and C++ programs
make	Maintains program source code
patch	Updates source code

Table 8-5: Programming utilities

As UNIX, and particularly Linux, gains market share, and applications continue to expand, use of the source code management utilities, listed in Table 8-6, should increase. These utilities have a proven track record in managing teamwork programming and are vital tools for scheduling and managing large-scale applications.

Utility	Brief Description of Function
ci	Creates changes in Revision Control Systems (RCS)
co	Retrieves an unencoded revision of an RCS file
cvs	Manages concurrent access to files in a hierarchy
rcs	Creates or changes the attributes of an RCS file
rlog	Prints a summary of the history of an RCS file

Table 8-6: Source code management utilities

Finally, miscellaneous utilities include unique programs that perform very specific and special functions. As you can see from the functions descriptions in Table 8-7, these commands include providing a system calendar, scheduling events, and identifying terminals attached to the system.

Utility	Brief Description of Function
at	Executes a shell script at a specified time
cal	Displays a calendar for a month or year
crontab	Schedules a command to run at a preset time
expr	Evaluates expressions (used for arithmetic and string manipulations)
fsck	Checks and fixes problems on a file system (repairs damages)
tee	Clones output stream to one or more files
tr	Replaces specified characters, like a translation filter
tty	Displays terminal path name, like the built-in command, <code>pwd</code>
xargs	Converts standard output of one command into arguments for another

Table 8-7: Miscellaneous utilities

Now that you understand that many diverse utilities are available, you are ready to use several of them to complete your tasks for Dominion Consulting.

Using the File Processing Utilities

Recall that the file processing utilities help you display and manipulate files. You use several of these utilities to complete your tasks. First, you use the `dd` command to copy a file and change the format of the destination file at the same time.

Using the `dd` Command

Files not only store information, but they also store it in a particular format. For example, most computers store text using ASCII codes. (IBM mainframes, however, use EBCDIC codes to store text.) In addition to the internal codes that computers use to store information, some files store text in all uppercase letters. Likewise, other files store text in all lowercase letters. Some files include only records, where each record consists of several fields. A special character, such as a colon, separates the fields, and each record ends with a new-line character. Different files have different internal formats, which depend upon how the file is used.

Whereas the standard UNIX copy utility, `cp`, duplicates a file, it cannot alter the format of the destination copy. Therefore, when you need to copy a file and change the format of the destination copy, use the `dd` command instead. Possessing a rich set of options that allow it to handle copies when other methods fail, the `dd` command can handle conversions to and from IBM's EBCDIC character types to the standard ASCII characters on non-IBM machines. The `dd` command is frequently used for devices such as tapes, which have discrete record sizes, or for fast multi-sector reads from disks.

Note: For a more complete description of the options available for this command and others in this chapter, refer to Appendix B, "Syntax Guide to UNIX Commands."

The `dd` command has the general form:

Syntax

`dd [options]`

Options include `if=filename` and `of=filename`, which specify the input files and output files, respectively. Another frequently used option is `bs=n`, which specifies the block size, *n*, as an integer. You also have options to specify the input block size (`ibs=n`) and output block size (`obs=n`). Specifying block size, an optional requirement, speeds copying, especially when copying backups to tape.

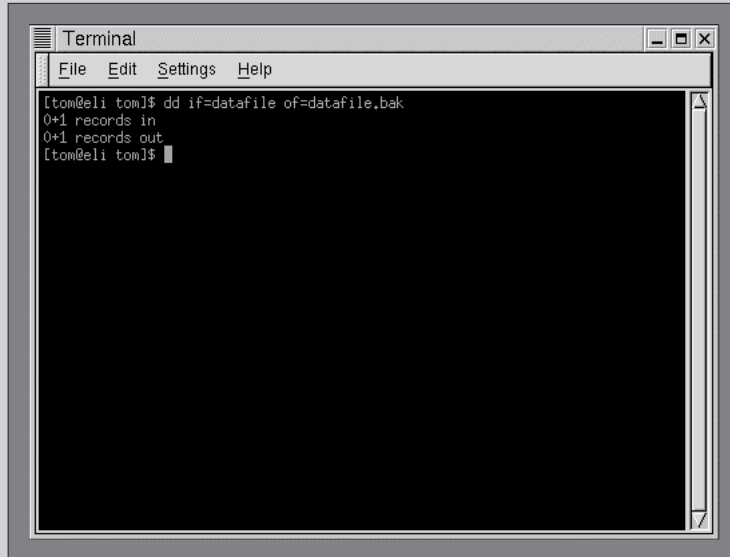
Another advantage that the `dd` command has over `cp` is that all users, not just the system administrator, can copy files to and from the floppy drive. With `cp`, you must mount a floppy disk if you need to copy to or from the floppy disk. Usually, only the system administrator, who must be logged on as root, can issue the mount command. With the `dd` command there is no need to mount the floppy device to access it.

By duplicating a file with `dd`, you can learn the command's basic usage.

To make a backup of a file using the dd command:

- 1 Use the cat command, or the editor of your choice, to create a file, datafile. The file should contain the following text:

This is my data file.
- 2 Make a copy of the file by typing **dd if=datafile of=datafile.bak** and pressing **Enter**. Your screen looks similar to Figure 8-1.



```
Terminal
File Edit Settings Help
[tom@eli tom]$ dd if=datafile of=datafile.bak
0+1 records in
0+1 records out
[tom@eli tom]$
```

Figure 8-1: Results of dd command

Making a Bootable Floppy Disk

It is a good idea to make a bootable floppy disk, because a computer problem (such as a crashed hard disk) may prevent you from starting UNIX from the system. To make a bootable floppy (in this case, for Red Hat Linux), you use four utility programs: the rdev, mkfs, fdformat, and dd commands. The rdev command queries and sets the **root device**, which is the hard disk partition that houses UNIX's root file system. The root file system, in turn, houses the UNIX kernel (core operating system), which is required to boot UNIX. (Under Red Hat Linux, the kernel is stored in a file whose name starts with vmlinuz. The file is usually stored in the /boot directory.) You can use the rdev command to set the root device with two

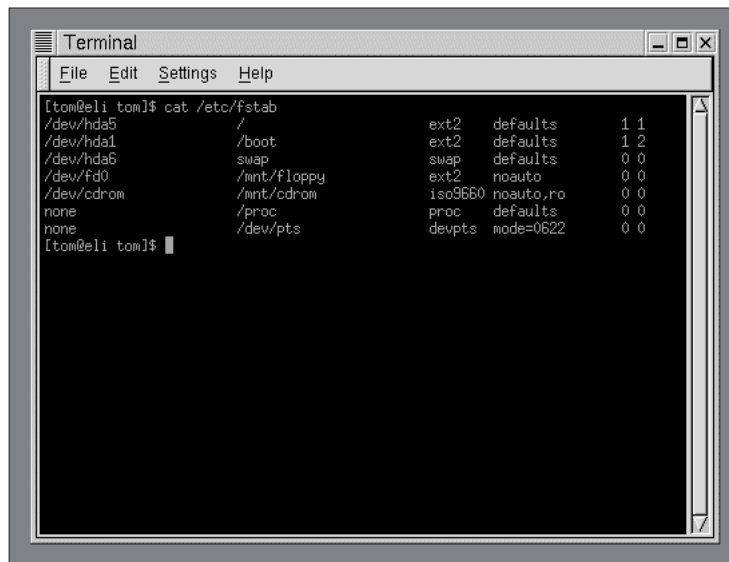
arguments, where *image* is the name of the file holding the kernel and *root device* is the name of the UNIX partition that holds the root file system:

Syntax `rdev [image] [root device]`

Use the `mkfs` command to build a UNIX or Linux file system on a device such as a floppy disk or a hard disk partition. The command is usually issued as follows, where `-t option` identifies the file system type and *filesystem* identifies the device name:

Syntax `mkfs [-t option] [filesystem]`

The file system type, which determines how UNIX reads and writes information from or onto the device, is stored in the file `/etc/fstab` (file system table), as shown in Figure 8-2.

A terminal window titled "Terminal" with a menu bar (File, Edit, Settings, Help). The command `cat /etc/fstab` has been executed, displaying the following text:

```
[tom@eli tom]$ cat /etc/fstab
/dev/hda5      /              ext2    defaults    1 1
/dev/hda1      /boot          ext2    defaults    1 2
/dev/hda6      swap           swap    defaults    0 0
/dev/fd0       /mnt/floppy    ext2    noauto      0 0
/dev/cdrom     /mnt/cdrom     iso9660 noauto,ro   0 0
none          /proc          proc    defaults    0 0
none          /dev/pts       devpts  mode=0622   0 0
[tom@eli tom]$
```

Figure 8-2: File system types as shown in `/etc/fstab`

The `fdformat` command formats a floppy at low levels to ensure that the media is clear of defects and is writeable. Unlike a high-level format that checks and verifies the disk's recording surface and sets up the file allocation tables (file

system), as done on a Microsoft operating system, the low-level format prepares the recording surface but does not set up a file system. (The latter is the function of the `mkfs` utility, so the two programs, `fdformat` and `mkfs`, must work in tandem to prepare the floppy disk for use.) The low-level format checks the recording surface more thoroughly than the high-level format does. The command-line entry to format a floppy follows this usage, where the `-n` option disables the verification performed after the format. The *filesystem* is usually indicated as `/dev/fd0` for drive A (the first floppy drive) and `/dev/fd1` for drive B (the second floppy drive):

Syntax

```
fdformat [-n] [filesystem]
```

Note: The default format type is for 1.44 megabyte (high-density) floppies. The *filesystem* may also be indicated as `/dev/fd0H1440` for a high-density floppy on drive A and `/dev/fd1H1440` for a high-density floppy on drive B.

You are now ready to make a bootable disk.

Note: The next steps make the following assumptions:

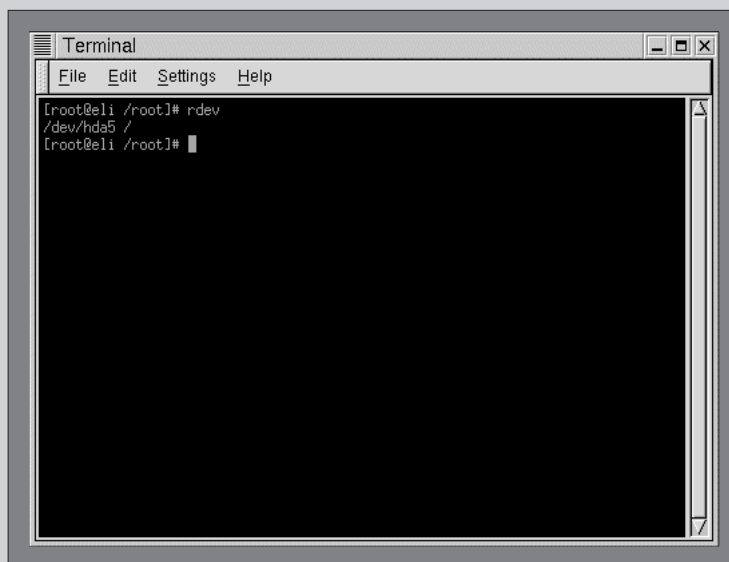
- You have system-level privileges (are logged on under the root account).
- You are logged on to the host console. You cannot perform the steps if you are accessing a Linux computer via a telnet session.

To make a bootable disk:

- 1** Determine where `vmlinuz`, the file holding the Linux kernel, is located (most likely in `/boot`).

If you have trouble locating the file, use the `find` command: type `find / -name vmlinuz*` and press **Enter**.
- 2** Verify the `/boot` partition's location by typing `rdev` and then pressing **Enter**. Normally, the boot partition is located on the `hda5` partition. Your screen should look similar to Figure 8-3.

help

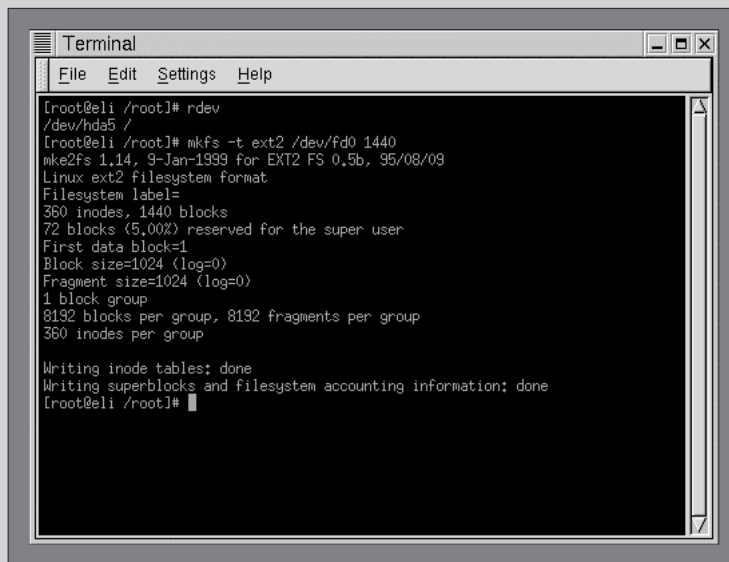
A terminal window titled "Terminal" with a menu bar (File, Edit, Settings, Help). The command prompt shows the user is root@eli in the directory /root. The command `rdev` has been executed, resulting in the output `/dev/hda5 /`.

```
Terminal
File Edit Settings Help

[root@eli /root]# rdev
/dev/hda5 /
[root@eli /root]#
```

Figure 8-3: Results of rdev command

- 3 Insert a blank floppy disk in drive A.
- 4 Make a Linux file system on disk by typing `mkfs -t ext2 /dev/fd0 1440` and then pressing **Enter**. Your screen should now look similar to Figure 8-4.

A terminal window titled "Terminal" with a menu bar (File, Edit, Settings, Help). The command prompt shows the user is root@eli in the directory /root. The command `mkfs -t ext2 /dev/fd0 1440` has been executed, resulting in a detailed output showing the creation of an ext2 filesystem. The output includes information about the number of inodes, blocks, reserved blocks, block size, fragment size, and block group, as well as the completion of writing inode tables and superblocks.

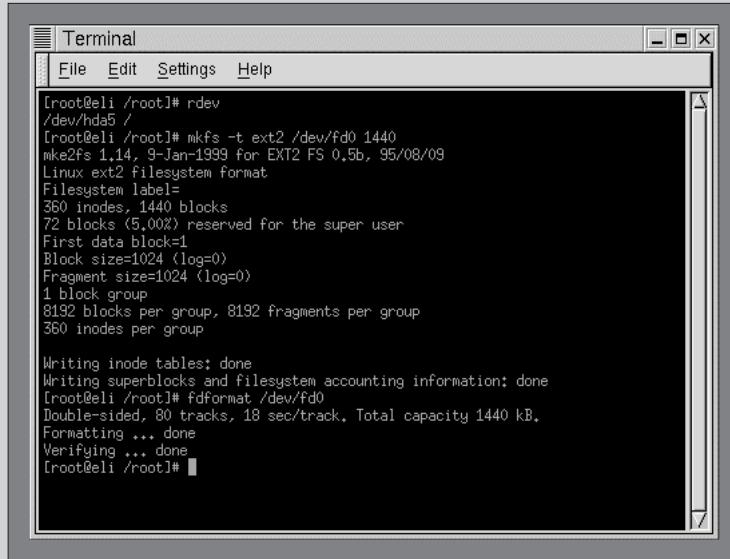
```
Terminal
File Edit Settings Help

[root@eli /root]# rdev
/dev/hda5 /
[root@eli /root]# mkfs -t ext2 /dev/fd0 1440
mke2fs 1.14, 9-Jan-1999 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
[root@eli /root]#
```

Figure 8-4: Results of mkfs command

- 5** Low-level format the disk by typing **fdformat /dev/fd0** and then pressing **Enter**. The utility displays its progress as it performs its operation. When the format is complete, your screen should look similar to Figure 8-5.



```
[root@eli /root]# rdev
/dev/hda5 /
[root@eli /root]# mkfs -t ext2 /dev/fd0 1440
mke2fs 1.14, 9-Jan-1999 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
[root@eli /root]# fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track, Total capacity 1440 kB.
Formatting ... done
Verifying ... done
[root@eli /root]#
```

Figure 8-5: Results of fdformat command

- 6** Make the floppy disk bootable by typing **dd if=/boot/vmlinuz-2.2.5-15 of=/dev/fd0** and then pressing **Enter**.

Note: If you are not using Linux Red Hat Version 6, the kernel image (in this case, vmlinuz-2.2.5-15) will differ. Use the one you located in Step 1.

Note: After you create a bootable disk, you can use it to start the system. Place the bootable disk in drive A, and turn on the system. Linux then boots from the floppy disk instead of the hard drive.

You can also use the dd command to make a back-up copy of a floppy disk. You can copy the entire contents of a floppy to a single file on your hard drive. You can then copy the file onto a second floppy disk, thus making a back-up copy of the original disk. In the next exercise, you make a back-up copy of the bootable floppy disk you created in the previous exercise. You need a second blank disk, which has already been formatted.



.....
To format a second floppy disk, follow Steps 3 through 5 in the previous exercise.
.....

Note: Like the steps in the previous exercise, the following steps can only be performed while you are logged on to the host console. The steps cannot be performed if you are accessing a Linux computer via a telnet session.

To make a back-up copy of the bootable floppy disk:

- 1** Insert the bootable floppy disk in the computer's floppy disk drive (commonly known as drive A).
- 2** Copy the disk's contents into your current directory on the hard drive by typing **dd if=/dev/fd0 > duplicate.floppy** and then pressing **Enter**.
- 3** Remove the floppy from drive A.
- 4** Insert a formatted floppy disk in drive A.
- 5** Copy the duplicate floppy disk image, **duplicate.floppy**, to the floppy by typing **dd if=./duplicate.floppy of=/dev/fd0** and then pressing **Enter**. Make sure to use the dot (.) reference to the input file (**if=./duplicate.floppy**) to indicate that the input file is located in the current directory.

You have learned some useful utilities for preparing floppy disks and copying information to them. Next you learn to monitor the hard disk usage of your system.

Checking Hard Disk Usage

UNIX system users, as well as the system itself, create and enlarge files. Eventually, unless files are removed, even the largest disk runs out of free space. To maintain adequate free space, you should use these basic strategies:

- Be vigilant against running dangerously low on free space by using the **df** command.
- Watch for conspicuous consumption by using the **du** command.
- Follow a routine schedule for “garbage” collection and removal by using the **rm** command.

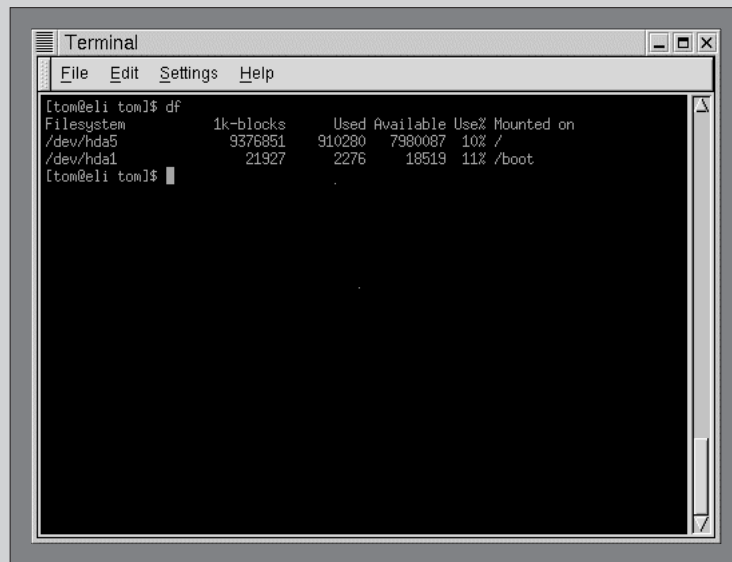
Now you use each of these techniques to check hard disk usage.

Using the df Utility

The df utility reports the number of 1024-byte blocks that are allocated, used, and available; the percentage used; and the mount point. The reports displayed are based on the command options entered.

To use the df command to check hard disk usage:

- 1 Type **df** and then press **Enter**. (Note that you can enter the df utility without options.) Your screen looks similar to Figure 8-6.

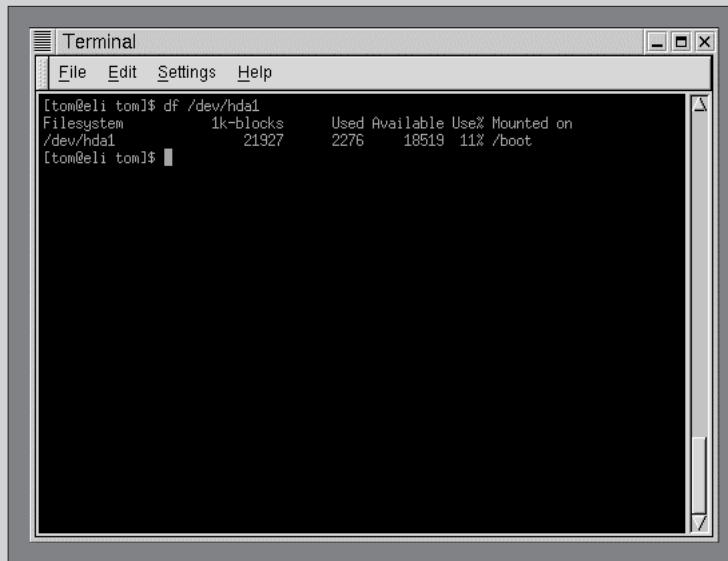
A terminal window titled "Terminal" with a menu bar (File, Edit, Settings, Help) and standard window controls. The terminal shows the command [tom@eli tom]\$ df and its output. The output is a table with columns: Filesystem, 1k-blocks, Used, Available, Use%, and Mounted on. The data rows are: /dev/hda5 (9376851 blocks, 910280 used, 7980087 available, 10% used, mounted on /) and /dev/hda1 (21927 blocks, 2276 used, 18519 available, 11% used, mounted on /boot). The prompt [tom@eli tom]\$ is shown at the bottom.

```
[tom@eli tom]$ df
Filesystem      1k-blocks    Used Available Use% Mounted on
/dev/hda5       9376851    910280  7980087   10% /
/dev/hda1        21927      2276   18519    11% /boot
[tom@eli tom]$
```

Figure 8-6: Output of df command

Of course, your file systems and their statistics will differ from those shown in Figure 8-6. In Figure 8-6, the df command reports that the /dev/hda5 file system has 9,376,851 blocks of 1 kilobyte each. There are 910,280 blocks in use, and 7,980,087 blocks available. Ten percent of the blocks are in use, and the file system is mounted on /.

- 2 You may specify a file system as an argument. The statistics for that file system alone appear on the screen. Type **df /dev/hda1** and press **Enter**. You see the disk statistics for the /dev/hda1 file system. Figure 8-7 shows an example.

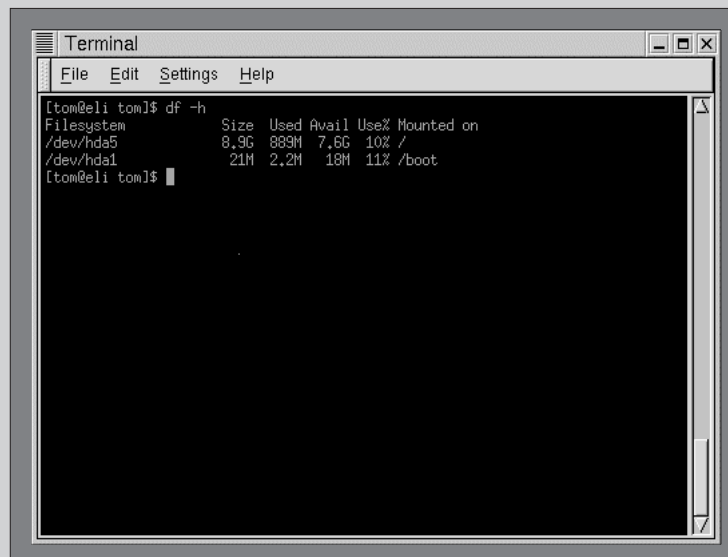


```
Terminal
File Edit Settings Help

[tom@eli tom]$ df /dev/hda1
Filesystem      1k-blocks      Used Available Use% Mounted on
/dev/hda1        21927         2276      18519   11% /boot
[tom@eli tom]$
```

Figure 8-7: Results of df command

- 3 The `-h` option causes the numbers to print in “human-readable” form. Instead of displaying raw numbers for size, amount of disk space used, and amount of space available, the statistics are printed in kilobyte, megabyte, or gigabyte format. Type `df -h` and press **Enter**. Figure 8-8 shows an example of the command’s output.



```
Terminal
File Edit Settings Help

[tom@eli tom]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/hda5        8.9G  889M   7.6G   10% /
/dev/hda1        21M   2.2M   18M   11% /boot
[tom@eli tom]$
```

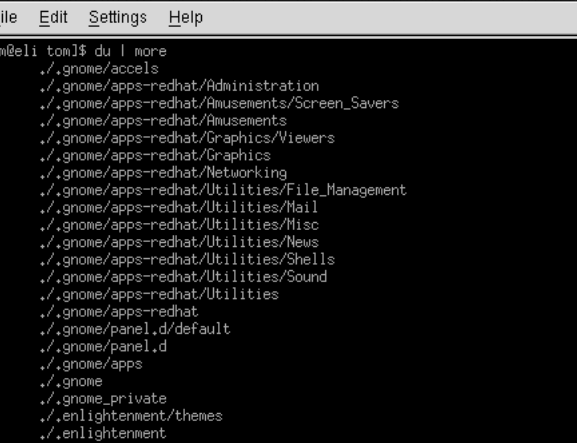
Figure 8-8: Output of `df -h` command

Using the du Utility

The `du` utility summarizes disk usage. If you enter the command without options, you receive a report based on all file usage, starting at your current directory and progressing down through all subdirectories. File usage is expressed in number of 512-byte blocks (default) or by the number of bytes (the `-b` option).

To report on disk usage:

- 1 If you are still logged on as root, log out and then log on under your account.
- 2 To receive a report on disk usage starting at your home directory, type **du | more** and then press **Enter**. (The results of the **du** command can be lengthy, so pipe its output to the **more** command.) Figure 8-9 shows an example of the command's output.

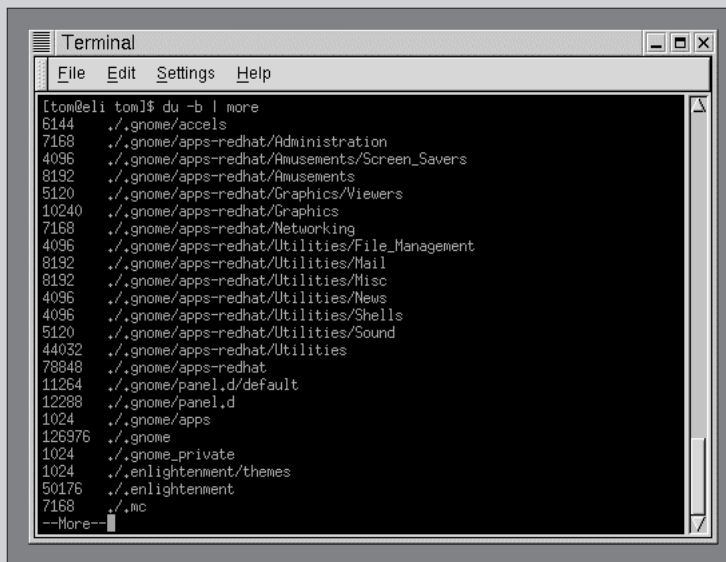


```
Terminal
File Edit Settings Help

[tom@elli tom]$ du -l | more
6      ./gnome/accels
7      ./gnome/apps-redhat/Administration
4      ./gnome/apps-redhat/Amusements/Screen_Savers
8      ./gnome/apps-redhat/Amusements
5      ./gnome/apps-redhat/Graphics/Viewers
10     ./gnome/apps-redhat/Graphics
7      ./gnome/apps-redhat/Networking
4      ./gnome/apps-redhat/Utilities/File_Management
8      ./gnome/apps-redhat/Utilities/Mail
8      ./gnome/apps-redhat/Utilities/Misc
4      ./gnome/apps-redhat/Utilities/News
4      ./gnome/apps-redhat/Utilities/Shells
5      ./gnome/apps-redhat/Utilities/Sound
43     ./gnome/apps-redhat/Utilities
77     ./gnome/apps-redhat
11     ./gnome/panel.d/default
12     ./gnome/panel.d
1      ./gnome/apps
124    ./gnome
1      ./gnome_private
1      ./enlightenment/themes
49     ./enlightenment
7      ./mc
--More--
```

Figure 8-9: Output of `du | more` command

- 3** The output shows the number of 512-byte blocks used in each subdirectory (including hidden subdirectories). Type **q** to exit the more command.
- 4** To receive a report on disk usage by number of bytes starting at your home directory, type **du -b | more** and then press **Enter**. Figure 8-10 shows an example of the command's output.



```

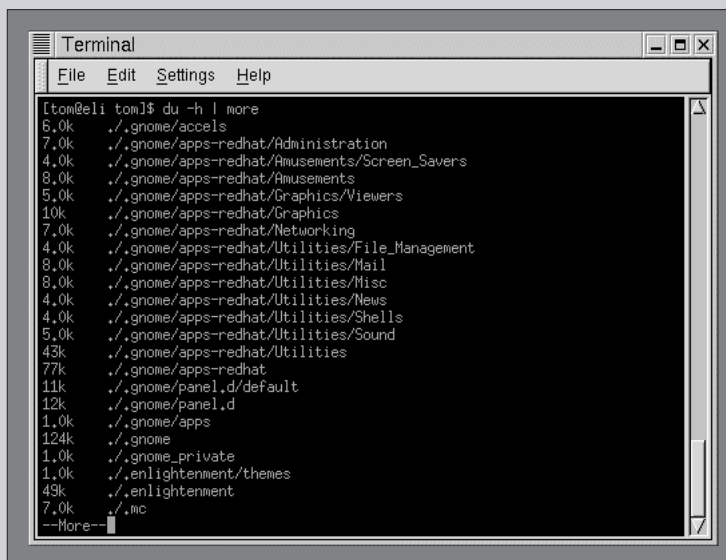
Terminal
File Edit Settings Help

[tom@eli tom]$ du -b | more
6144  ./,gnome/acceIs
7168  ./,gnome/apps-redhat/Administration
4096  ./,gnome/apps-redhat/Amusements/Screen_Savers
8192  ./,gnome/apps-redhat/Amusements
5120  ./,gnome/apps-redhat/Graphics/Viewers
10240 ./,gnome/apps-redhat/Graphics
7168  ./,gnome/apps-redhat/Networking
4096  ./,gnome/apps-redhat/Utilities/File_Management
8192  ./,gnome/apps-redhat/Utilities/Mail
8192  ./,gnome/apps-redhat/Utilities/Misc
4096  ./,gnome/apps-redhat/Utilities/News
4096  ./,gnome/apps-redhat/Utilities/Shells
5120  ./,gnome/apps-redhat/Utilities/Sound
44032 ./,gnome/apps-redhat/Utilities
78848 ./,gnome/apps-redhat
11264 ./,gnome/panel.d/default
12288 ./,gnome/panel.d
1024  ./,gnome/apps
126976 ./,gnome
1024  ./,gnome_private
1024  ./,enlightenment/themes
50176 ./,enlightenment
7168  ./,mc
--More--

```

Figure 8-10: Output of `du -b | more` command

- 5 Type **q** to exit the more command.
- 6 Like the `df` command, the `du` command supports the `-h` option to display statistics in human-readable format. Type `du -h | more` and press **Enter**. Figure 8-11 shows an example of the command's output.



```

Terminal
File Edit Settings Help

[tom@eli tom]$ du -h | more
6.0k  ./,gnome/acceIs
7.0k  ./,gnome/apps-redhat/Administration
4.0k  ./,gnome/apps-redhat/Amusements/Screen_Savers
8.0k  ./,gnome/apps-redhat/Amusements
5.0k  ./,gnome/apps-redhat/Graphics/Viewers
10k   ./,gnome/apps-redhat/Graphics
7.0k  ./,gnome/apps-redhat/Networking
4.0k  ./,gnome/apps-redhat/Utilities/File_Management
8.0k  ./,gnome/apps-redhat/Utilities/Mail
8.0k  ./,gnome/apps-redhat/Utilities/Misc
4.0k  ./,gnome/apps-redhat/Utilities/News
4.0k  ./,gnome/apps-redhat/Utilities/Shells
5.0k  ./,gnome/apps-redhat/Utilities/Sound
43k   ./,gnome/apps-redhat/Utilities
77k   ./,gnome/apps-redhat
11k   ./,gnome/panel.d/default
12k   ./,gnome/panel.d
1.0k  ./,gnome/apps
124k  ./,gnome
1.0k  ./,gnome_private
1.0k  ./,enlightenment/themes
49k   ./,enlightenment
7.0k  ./,mc
--More--

```

Figure 8-11: Output of `du -h | more` command

Removing Garbage Files

An easy way to free space in your file systems is to remove garbage files. **Garbage files** are temporary files, such as a core file, that lose their usefulness after several days. A **core file** is created when an executing program attempts to do something illegal, like accessing another user's memory. The UNIX operating system detects the attempt and sends a signal to the program. The signal halts the offending program and creates a copy of the program and its environment in a file named `core` in the current directory. The programmer who wrote the program that “dumps core” (slang for this event) might be interested in dissecting the core file with a debugging tool. However, all too often the core file simply languishes unused in some branch of the directory hierarchy. All files created this way have the same name: `core`.

Another file with a generic name is `a.out`, the default for the output of program compilation procedures. Like core files, the true identity of these generically named files often gets lost over time. You can use a pipeline construction of the `find` command to retrieve these wasteful files and execute the `rm` command to remove them. You can also use the `find` command to search through the directory hierarchy and remove such files. You can write a `find` command so that it locates all files named `core` and `a.out`, and then remove each one. Here is such a command:

```
find . "(" -name a.out -o -name core ")" -exec rm {} \;
```

You have used the `find` command before to locate files. The command above locates every occurrence of the `a.out` and `core` files, and then deletes them with the `rm` command. The first argument, `dot` (`.`), tells `find` to start looking in the current directory. The argument “(`-name a.out -o -name core`)” uses the `-o` (OR) operator. It tells `find` to search for files named `a.out` OR `core`. The `-exec rm` option instructs `find` to execute the `rm` command each time it locates a file with the name being searched for. The `{}` characters are replaced with the matching file name. For example, when the command locates an `a.out` file, the `{}` are replaced with `a.out`, so the command `rm a.out` is executed. The `;` terminates the command.

Now you use the `find` command to search for and delete all occurrences of `a.out` and `core`.

Note: The steps that follow assume you have a source directory in your home directory. The test files, `a.out` and `core`, are quickly created using the `touch` command, which, when followed by the filenames, creates empty files. The tilde (`~`) ensures that these files go into your home and source directories.

To remove garbage files:

- 1 Create some garbage files, `core` and `a.out`, and place them in your home directory and the subdirectory of home by typing `touch ~/core`, pressing **Enter**, typing `touch ~/a.out`, pressing **Enter**, typing `touch ~/source/core`, pressing **Enter**, typing `touch ~/source/a.out` and then pressing **Enter**.

- 2 Make sure you are in your home directory, then type **find . "(" -name a.out -o -name core ")"** and press **Enter**. Your screen should resemble Figure 8-12.

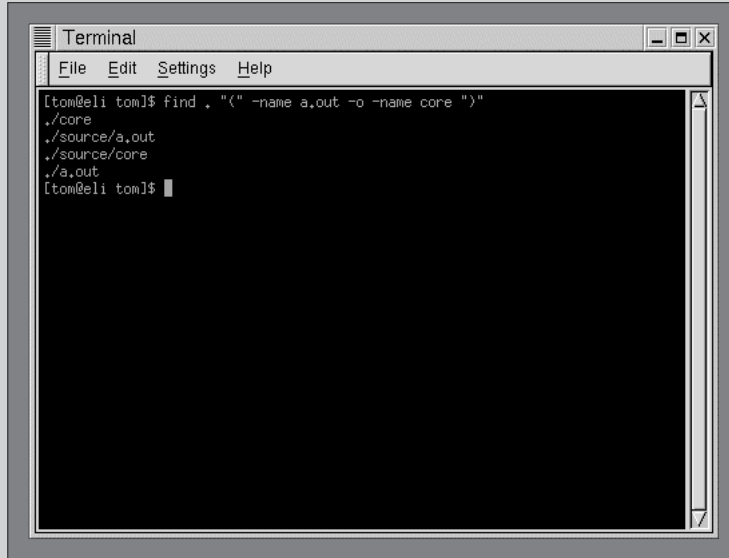


Figure 8-12: Results of find command

The results of the find command in Step 2 show the core and a.out files are in the current directory and in the source directory.

- 3 Remove the garbage files by typing **find . "(" -name a.out -o -name core ")" -exec rm{} \;** and then pressing **Enter**.
- 4 Check that the files have been removed by repeating the find command you entered in Step 2. If the files have been removed correctly, there will be no output. If the files still exist, you didn't enter the find command in Step 3 correctly. Retype the command exactly as it appears in Step 3, and then repeat this step (4).

You can locate several other garbage files with the find command. For example, users often name files test, temp, or tmp to indicate temporary files that may be forgotten over time and should be removed.

You have now had an opportunity to use several file processing utilities. Next, you use some system status utilities.

Applying System Status Utilities

As you see from the list of command descriptions in Table 8-2, the system status commands determine the system’s performance. Although system engineers who assess the CPU’s performance primarily use this data, you should at least know how to obtain this information. You can redirect the output of these commands to a file that you can then print and forward to the system administrator and system tune-up specialists.

One of the most effective utilities for auditing system performance is the `top` command. The `top` command displays a listing of the most CPU-intensive tasks, such as the processor state, in real time (the display is updated every 5 seconds by default). This means that you can actually see what is happening inside the computer as it progresses. The `top` command works with these options:

Syntax	<code>top [-] [d delay] [q] [S] [s] [i] [c]</code>
Dissection	The <code>d</code> option specifies the delay between screen updates. The <code>q</code> option causes the <code>top</code> utility to refresh without delay. The <code>S</code> option specifies cumulative mode, where each process is listed with the CPU time that it has spent. The <code>s</code> option allows the <code>top</code> utility to run in secure mode, which disables the interactive commands (a good option for those not in charge of tuning the system). The <code>i</code> option causes the <code>top</code> utility to ignore any idle processes. Finally, the <code>c</code> option displays the command line instead of the command name only. While running, the <code>top</code> command supports interactive commands, such as <code>k</code> , which kills a running process. The <code>top</code> utility continues to produce output until you press <code>q</code> to terminate the execution of the program.

The simplest way for most users to run the `top` utility is to simply issue the command without options.

To use the `top` utility:

- 1 Display the CPU activity by typing `top` and then pressing **Enter**. Your screen should look similar to Figure 8-13. (Don’t forget this display changes dynamically while on screen.)
- 2 The processes are shown in the order of the amount of CPU time they use. After looking at the display for a short time, press **q** to exit from the `top` utility.
- 3 Run the `top` command again. Notice the leftmost column of information, labeled PID. This column lists the process ID of each process shown. Notice the PID of the `top` command. (In Figure 8-13, the `top` command’s PID is 707. Yours will probably differ.)

```

5:03pm up 1:24, 2 users, load average: 0.00, 0.06, 0.06
57 processes: 55 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 2.3% user, 1.7% system, 0.0% nice, 95.8% idle
Mem: 63140K av, 61456K used, 1684K free, 42064K shrd, 4256K buff
Swap: 68004K av, 80K used, 67924K free, 33980K cached

  PID USER   PRI  NI  SIZE  RSS SHARE STAT   LIB %CPU %MEM    TIME COMMAND
  578 root    18   0 10744  10M 2064 R       0  1.5 17.0   0:19 X
  708 tom     17   0 2092 2092 1720 S       0  0.9  3.3   0:00 screenshot
  707 tom     16   0 1008 1008  824 R       0  0.5  1.5   0:00 top
  599 tom      3   0 2276 2276 1608 S       0  0.3  3.6   0:01 enlightenmen
  614 tom      1   0 1656 1656 1228 S       0  0.1  2.6   0:00 xscreensaver
  630 tom      3   0 2980 2980 2404 S       0  0.1  4.7   0:00 gnomepager_a
  642 tom     15   0 6340 6340 2676 S       0  0.1 10.0   0:02 gimp
    1 root      0   0   476   476   408 S       0  0.0  0.7   0:03 init
    2 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kpiod
    4 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kswapd
    5 root     -20 -20     0     0     0 SWK     0  0.0  0.0   0:00 mdrecoveryd
  102 root      0   0   460   460   400 S       0  0.0  0.7   0:00 apmd
  191 bin      0   0   376   376   308 S       0  0.0  0.5   0:00 portmap
  238 root      0   0   592   592   492 S       0  0.0  0.9   0:00 syslogd
  249 root      0   0   752   752   388 S       0  0.0  1.1   0:00 klogd
  263 daemon    0   0   472   472   400 S       0  0.0  0.7   0:00 atd
  277 root      0   0   592   592   504 S       0  0.0  0.9   0:00 crond

```

Figure 8-13: Output of top command

- 4 Press **k** to initiate the kill command. The top program asks you to enter the PID to kill. Enter the PID of the top command. Your screen should resemble Figure 8-14. Press **Enter** to kill the process. (You may have to press Enter a second time to return to a command prompt.) The top command is no longer running.

```

5:03pm up 1:25, 2 users, load average: 0.10, 0.08, 0.07
56 processes: 54 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 1.9% user, 0.5% system, 0.0% nice, 97.4% idle
Mem: 63140K av, 61196K used, 1944K free, 40348K shrd, 4272K buff
Swap: 68004K av, 80K used, 67924K free, 33980K cached
PID to kill: 707

  PID USER   PRI  NI  SIZE  RSS SHARE STAT   LIB %CPU %MEM    TIME COMMAND
  578 root    19   0 10736  10M 2064 R       0  0.7 17.0   0:20 X
  707 tom      8   0 1008 1008  824 R       0  0.5  1.5   0:00 top
  599 tom      3   0 2276 2276 1608 S       0  0.3  3.6   0:01 enlightenmen
  613 tom      1   0 4164 4164 2808 S       0  0.1  6.5   0:00 panel
  630 tom      2   0 2980 2980 2404 S       0  0.1  4.7   0:00 gnomepager_a
  636 tom      1   0 3056 3056 2432 S       0  0.1  4.8   0:03 gnome-termin
  642 tom     11   0 6464 6464 2680 S       0  0.1 10.2   0:02 gimp
    1 root      0   0   476   476   408 S       0  0.0  0.7   0:03 init
    2 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kpiod
    4 root      0   0     0     0     0 SW      0  0.0  0.0   0:00 kswapd
    5 root     -20 -20     0     0     0 SWK     0  0.0  0.0   0:00 mdrecoveryd
  102 root      0   0   460   460   400 S       0  0.0  0.7   0:00 apmd
  191 bin      0   0   376   376   308 S       0  0.0  0.5   0:00 portmap
  238 root      0   0   592   592   492 S       0  0.0  0.9   0:00 syslogd
  249 root      0   0   752   752   388 S       0  0.0  1.1   0:00 klogd
  263 daemon    0   0   472   472   400 S       0  0.0  0.7   0:00 atd
  277 root      0   0   592   592   504 S       0  0.0  0.9   0:00 crond

```

Figure 8-14: The k command

- 5 Run the top utility in secure mode by typing **top -s** and pressing **Enter**.
- 6 Press **k** to initiate the kill command. Because top is running in secure mode, it displays the message “Can’t kill in secure mode,” as shown in Figure 8-15.

```

Terminal
File Edit Settings Help

5:11pm up 1:33, 2 users, load average: 0.00, 0.01, 0.02
56 processes: 54 sleeping, 2 running, 0 zombie, 0 stopped
CPU states: 1.8% user, 0.7% system, 0.0% nice, 97.4% idle
Mem: 63140K av, 60992K used, 2148K free, 39560K shrd, 4252K buff
Swap: 68004K av, 176K used, 67828K free
Can't kill in secure mode
  PID USER   PRI NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM  TIME COMMAND
    578 root    15  0 10784 10M 2068 R   0 1.2 17.0 0:22 X
    599 tom      4  0 2276 2276 1608 S   0 0.3 3.6 0:01 enlightenmen
    630 tom      3  0 2984 2984 2404 S   0 0.2 4.7 0:01 gnomepager_a
    642 tom      8  0 6464 6464 2680 S   0 0.1 10.2 0:03 gimp
    727 tom      5  0 1024 1024 840 R   0 0.1 1.6 0:00 top
      1 root     0  0 476 476 408 S   0 0.0 0.7 0:03 init
      2 root     0  0 0 0 0 SW  0 0.0 0.0 0:00 kflushd
      3 root     0  0 0 0 0 SW  0 0.0 0.0 0:00 kpiod
      4 root     0  0 0 0 0 SW  0 0.0 0.0 0:00 kswapd
      5 root    -20 -20 0 0 0 SW< 0 0.0 0.0 0:00 mdrecoveryd
    102 root     0  0 384 380 324 S   0 0.0 0.6 0:00 apmd
    191 bin      0  0 376 376 308 S   0 0.0 0.5 0:00 portmap
    238 root     0  0 540 540 440 S   0 0.0 0.8 0:00 syslogd
    249 root     0  0 676 668 312 S   0 0.0 1.0 0:00 klogd
    263 daemon    0  0 296 284 224 S   0 0.0 0.4 0:00 std
    277 root     0  0 532 532 444 S   0 0.0 0.8 0:00 crond
    290 root     0  0 492 484 372 S   0 0.0 0.7 0:00 cardmgr
    304 root     0  0 508 508 424 S   0 0.0 0.8 0:00 inetd
  
```

Figure 8-15: Top utility in secure mode

- 7 Press **q** to exit the top utility.

Table 8-8 summarizes several of the top command’s options.

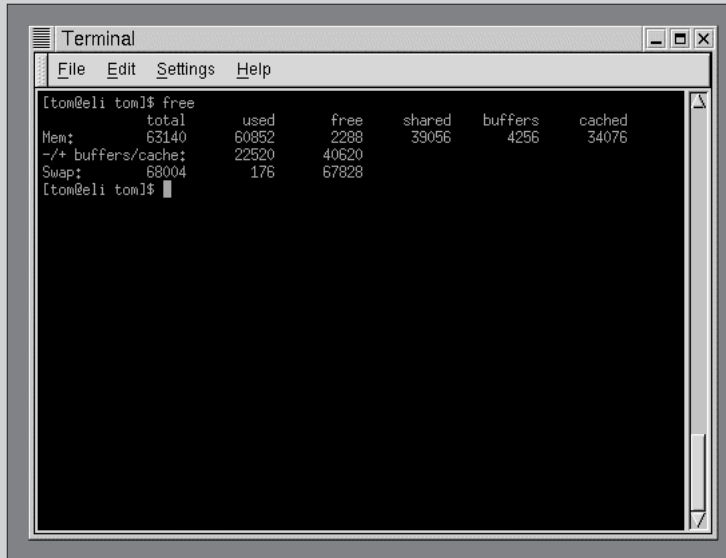
Option	Description
-q	Causes the top command to display its output continually, with no delay between outputs
-s	Causes top command to run in secure mode, disabling its interactive commands, such as k (for kill)
-S	Runs top in cumulative mode. This mode displays the cumulative CPU time used by a process instead of the current CPU time used.
-l	Causes top to ignore any idle processes
-c	Causes top to display the command line that initiated each process, instead of only displaying the program name

Table 8-8: Options for top command

Another useful, though static, display of memory usage is generated by the **free** command. The **free** command displays the amount of free and used memory in the system. Unlike **top**, the **free** utility runs and then automatically exits.

To demonstrate the free command:

- 1 Type **free** and press **Enter**. Your screen looks similar to Figure 8-16.



```
[tom@eli tom]$ free
              total        used        free      shared    buffers     cached
Mem:          63140        60852         2288       39056        4256       34076
~/+ buffers/cache: 22520        40620
Swap:        68004           176       67828
[tom@eli tom]$
```

Figure 8-16: Output of **free** command

- 2 The command displays the amount of total, used, and free memory. It also displays the amount of shared memory, buffer memory, and cached memory. In addition, the amount of total, used, and free swap memory is shown. By default, all amounts are shown in kilobytes.
- 3 Type **free -m** and press **Enter** to see the **free** command's output in megabytes.
- 4 Type **free -b** and press **Enter** to see the **free** command's output in bytes.

As mentioned earlier, you may want to forward these displays to the system administrator and tune-up specialists. You do this next.

To forward displays generated by the top and free utilities:

- 1 Redirect the outputs of the **top** utility to a file in your current directory by typing **top > top_out** and then pressing **Enter**.

- 2 Wait about 10 seconds and then press **q** to exit the top utility.
- 3 Redirect the outputs of the top free to a file in your current directory by typing **free > free_out** and then pressing **Enter**.
- 4 Print the report for the system administrator and tune-up specialists by typing **lpr top_out**, pressing **Enter**, typing **lpr free_out**, and then pressing **Enter** again.

Because UNIX is a multitasking operating system, it allows you to run programs in the background while you continue to work with other programs. For example, if you have a program that prints a lengthy report, you can run it in the background and continue working with other programs while the report is printing. To run a program in the background, append the **&** character to the end of the command line.

To run a program in the background:

- 1 Experiment by running the top utility in the background. Type **top &** and press **Enter**.

The system reports the PID of the program that you started in the background. In Figure 8-17, the PID is 797. The top utility runs, but because it runs in the background, you see no output.

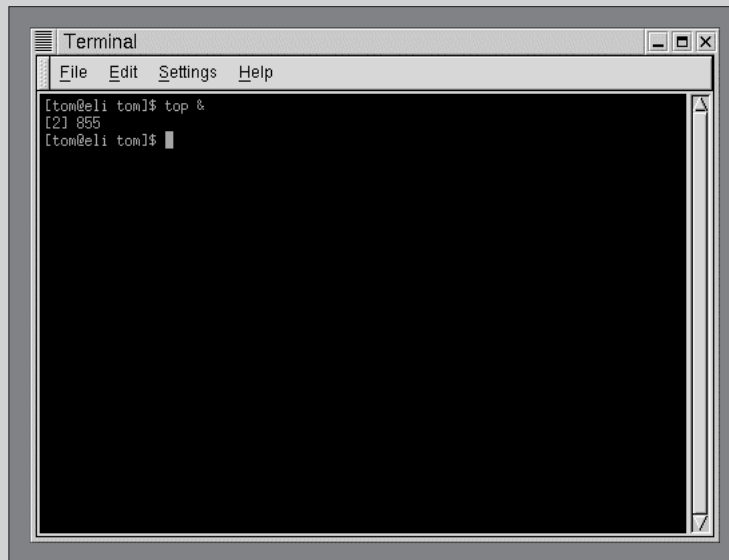


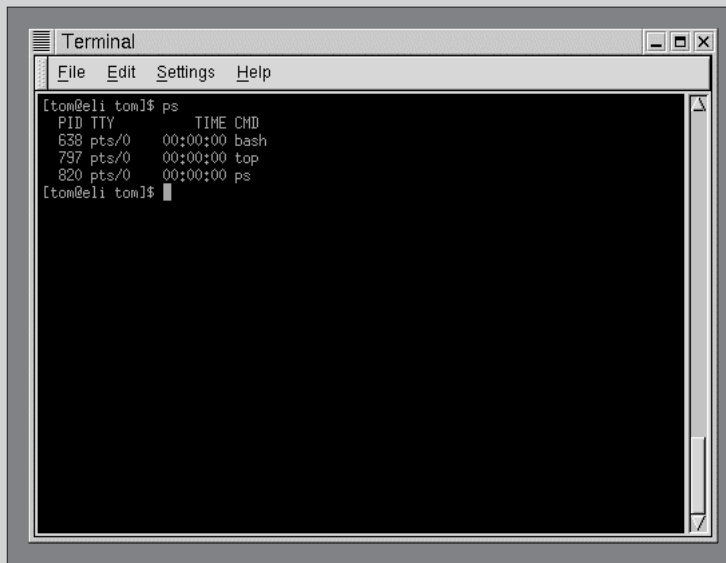
Figure 8-17: Running top utility in the background

- 2 Continue to run the top utility in the background. The process is the subject of the next two exercises.

The `ps` command shows you a list of the processes currently running. When you use the command with no options, it shows a list of the processes associated with the current login session. When used with the `-A` option, it shows a very long list of all processes running on the system.

To use the `ps` command:

- 1 Type `ps` and press **Enter**. Your screen looks similar to Figure 8-18.



```
Terminal
File Edit Settings Help
[tom@eli tom]$ ps
  PID TTY          TIME CMD
   638 pts/0    00:00:00 bash
   797 pts/0    00:00:00 top
   820 pts/0    00:00:00 ps
[tom@eli tom]$
```

Figure 8-18: `ps` command

- 2 The output of the `ps` utility includes this information about each process:
 - PID
 - Name of the terminal where the process started
 - Amount of time the process has been running
 - Name of the process

Notice the top utility still runs in the background.

- 3 To see a list of all processes running on the system, type `ps -A | more` and press **Enter**. Figure 8-19 shows an example of the command's output.

```

Terminal
File Edit Settings Help
[ton@eli ton]$ ps -A | more
PID TTY          TIME CMD
  1 ?            00:00:03 init
  2 ?            00:00:00 kflushd
  3 ?            00:00:00 kpiod
  4 ?            00:00:00 kswapd
  5 ?            00:00:00 mdrecoveryd
102 ?            00:00:00 apmd
191 ?            00:00:00 portmap
238 ?            00:00:00 syslogd
249 ?            00:00:00 klogd
263 ?            00:00:00 atd
277 ?            00:00:00 crond
290 ?            00:00:00 cardmgr
304 ?            00:00:00 inetd
318 ?            00:00:00 lpd
335 ?            00:00:00 rpc.statd
346 ?            00:00:00 rpc.rquotad
357 ?            00:00:00 rpc.mountd
372 ?            00:00:00 nfsd
373 ?            00:00:00 nfsd
374 ?            00:00:00 nfsd
375 ?            00:00:00 nfsd
376 ?            00:00:00 nfsd
--More--

```

Figure 8-19: ps -A command

- 4 Press the **spacebar** until the command finishes its output.

To force a process to terminate, use the kill command. Its two formats are:

Syntax

```
kill [process ID]
or...
kill %[process name]
```

When the kill command successfully executes, it terminates the process whose PID or name is passed as an argument.

To use the kill command:

- 1 In this exercise you terminate the top utility that is still running in the background. Type **ps** and press **Enter**. Look at the list of process to find the top utility's PID.
- 2 Type one of these commands (both perform the same operation):
kill <process id> and press **Enter**, or
kill %top and press **Enter**

- 3** Use the **ps** command again to see a list of the running processes. The top utility is no longer running.

Caution: Be very careful when using the kill command. If you kill a process that the operating system needs, you can cause disastrous results!

In this section, you learned about several utilities to monitor and maintain the system. You learned to use the **dd** command to copy files and duplicate floppy disks. You used the **mkfs** command to create a file system on a floppy disk and used the **fdformat** command to perform a low-level format on a floppy disk. You learned to query the partition containing the root device with the **rdev** command. Also, you learned to monitor disk, memory, and CPU usage with the **df**, **du**, **top**, and **free** commands. Finally, you learned to run programs in the background with the **&** operator, to list all running processes with the **ps** command, and to terminate processes no longer needed with the **kill** command. In the next section you learn about utilities that format text files.



S U M M A R Y

- UNIX utilities are classified into seven major function areas dictated by user needs: file processing, networking, communications, system status, programming, source code management, and miscellaneous tasks.
- Utility programs are distinguished from other operating system programs, because they are “add-ons” and not a part of the UNIX shells, nor a component of the “kernel.”
- Because utility programs are executed by entering their names on the command line, these programs are commonly referred to as commands.
- The **dd** command possesses a rich set of options that allow it to handle copies when other copying methods fail.
- To make a bootable floppy disk, you use four utility programs: **rdev**, **mkfs**, **fdformat**, and the **dd** commands.
- The **rdev** command queries and sets the root device. The root device is the hard disk partition that houses UNIX’s root file system.
- The **fdformat** command low-level formats a floppy to ensure that the media is without defect and is writeable to set up a file system.
- The **mkfs** utility sets up a file system and works in tandem with **fdformat** to prepare the floppy disk for use.
- The **df** utility checks and reports on free disk space.
- The **du** command checks for disk usage (consumption).

- You can use a pipeline construction of the `find` command to retrieve wasteful files and then execute the `rm` command to remove them from the hard disk.
- The `top` and `free` utilities provide detailed views of the “internals” of the system that are invisible to the naked eye but directly related to the CPU’s performance.
- You can redirect the output of the `top` and `free` commands to a disk file to use as input to print a report for the system administrator and system tune-up specialists.
- You run a program in the background by appending the `&` operator to the end of the command line.
- The `ps` command displays all processes currently running.
- The `kill` command terminates a specific process.



REVIEW QUESTIONS

1. UNIX utilities differ from other UNIX system programs in that _____.
 - a. they are built into the shells
 - b. they are an integral part of the kernel
 - c. they handle file processing tasks
 - d. they are external to the shells and the core operating system
2. The `rdev` command displays the _____.
 - a. root directory
 - b. root device
 - c. floppy disk device
 - d. user’s home directory
3. The `mkfs` command _____.
 - a. creates a file system type
 - b. sets the root device
 - c. makes a bootable floppy
 - d. formats the floppy
4. Use the _____ command to make sure you are not running out of free space on the disk.
 - a. `dd`
 - b. `du`
 - c. `df`
 - d. `rdev`
5. Use the _____ command to determine how much file space is being consumed.
 - a. `dd`
 - b. `du`
 - c. `df`
 - d. `rdev`

6. Use the _____ command to summarize how disk space is being used.
 - a. dd
 - b. du
 - c. df
 - d. rdev
7. Display the CPU activity using the _____ command.
 - a. top
 - b. free
 - c. test
 - d. review
8. Which command locates and displays the a.out and core files recursively starting at the current directory?
 - a. find . (“-name a.out -o -name core “)”
 - b. find / a.out -o -name core
 - c. find / -name “a.out” -o -name “core”
 - d. find ~/ -name “a.out” -o -name “core”
9. Which command terminates a running program or process?
 - a. abort
 - b. terminate
 - c. end
 - d. kill
10. Which option of the ps command displays all processes running in the system?
 - a. -a
 - b. -A
 - c. -p
 - d. -all
11. How do you terminate a process with the top utility?
 - a. with the k command
 - b. with the -a option
 - c. with the a command
 - d. with the -kill option



EXERCISES

1. Use the cat command or the editor of your choice to create a file, my_info. The file should contain your name and address. Use the dd command to copy the file to my_info.dup.
2. Insert a blank disk in the floppy drive of your computer. Use the mkfs command to create an ext2 file system on the disk.
3. Use the fdformat command to low-level format the disk in the floppy disk drive.
4. Use the dd command to copy the file my_info (the one you created in Exercise 1) to the floppy disk.

5. Use the `dd` command to make a duplicate of the floppy disk. Store the duplicate in a file `dup.floppy` in your home directory.
6. Use the `dd` command to copy the `dup.floppy` file (the one you created in Exercise 5) onto a second blank disk.
7. Use the redirection operator (`>`) to create test files `test1`, `test2`, and `test3` in your home directory, and copy them into the source directory. Use the `find` command to display all the files.
8. Use the `free` command to determine the amount of free system memory.
9. Run the `top` program in the background.
10. Use the `free` command to check the amount of free system memory again. Has it changed because you are now running a program in the background?
11. Use the `ps` command to determine the `top` program's PID.
12. Use the `kill` command to terminate the `top` program, which is still running in the background.



DISCOVERY EXERCISES

1. Use the `df` command with the correct option to display the number of gigabytes of disk space that are available on your system.
2. Use the `du` command to determine which subdirectory under your home directory holds the most information.
3. Use the `du` command again, but cause it to display its output in bytes rather than blocks.
4. Use the `du` command and specify the root directory (`/`) as its starting point. How many blocks does the `/dev` directory use?
5. In the section, "Removing Garbage Files," you entered a `find` command that locates and removes all files named `a.out` and `core`. Re-enter the same `find` command, but this time it should locate and remove the `test1`, `test2`, and `test3` files you created in Exercise 7.
6. Run the `man` program with the argument `du` in the background.
7. Run the `top` program. Is the `man` program, which is currently running in the background, listed? Why or why not?
8. Use the `ps` command to see the currently running processes. What is the PID of the `man` command that is still running?
9. Use the `kill` command to terminate the `man` program.

LESSON B

objectives

In this lesson you will:

- Check the spelling of a file's contents
- Create your own man page
- Use the **groff** utility to format a man page document
- Use **man** to display your new man page

Working with the Text Formatting File Utilities

Spell-checking a Document

Sometimes, the simpler the command, the more useful it is. This is the case with the **ispell** utility. The **ispell** utility scans a text document, displays errors on the screen, and suggests other words with similar spellings as replacements for unrecognized words. A menu that appears on the bottom line of the screen shows corrective options and exit codes.

To become more familiar with the **ispell** utility, you next type a sample document.

To see how the **ispell** utility works:

- 1 Use the **vi** or **Emacs** editor to create a file and name it **document1**.
- 2 Enter the following text, with misspellings:
This is a document that describes our newest and fastest machineery. Take the time to lern how to use each piece of equipment.
- 3 Save the file and exit the editor.
- 4 Scan the file for spelling errors by typing **ispell document1** and then pressing **Enter**. Your screen should look similar to Figure 8-20.

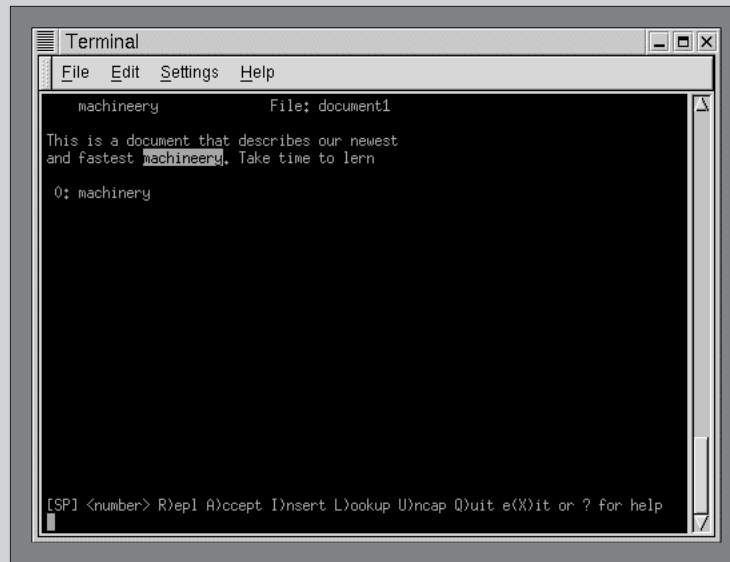


Figure 8-20: ispell utility

- 5 To correct the word “machinery,” look at the menu options at the bottom of the screen, and then type **0** (zero) to replace it with the correctly spelled word.
- 6 On the next screen, the next misspelled word is block-highlighted, as shown in Figure 8-21. Type **05** to replace “lern” with “learn.”

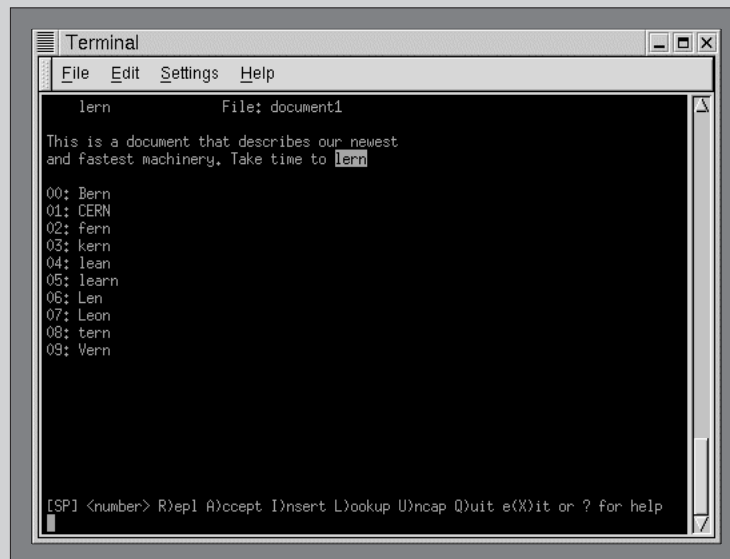


Figure 8-21: Correcting spelling mistakes with the ispell utility

Note: If the word “learn” appears next to a different number on your system, type that number.

- 7 The program exits and returns you to the command line. Type **cat document1** and press **Enter**.
- 8 The misspelled words have been corrected.

Now you learn about the `cmp` utility, which compares files and determines the first difference between them.

Comparing Files

Suppose you have a file that you work with regularly. You make a back-up copy of the file for safekeeping. Later, you want to see if the original file has changed since you made the back-up copy. You can use the `cmp` utility to compare the contents of two files and report the first difference between them.

The general form of the `cmp` command is:

Syntax	<code>cmp file1 file2</code>
---------------	------------------------------

The `cmp` command displays the character position and line number of the first difference between the two files.

To compare two files with the `cmp` command:

- 1 Use the `vi` or Emacs editor to create the file `file1`, containing the text:

```
This is file 1.  
I made it myself.  
It belongs to me.
```
- 2 Save the file.
- 3 Use the editor to create the file `file2`, containing the text:

```
This is file 2.  
I made it myself.  
It belongs to you.
```
- 4 Save the file and exit the editor.
- 5 At the command line, type **cmp file1 file2** and press **Enter**. Your screen looks similar to Figure 8-22. The command reports that the two files differ at character position 14 on line 1.

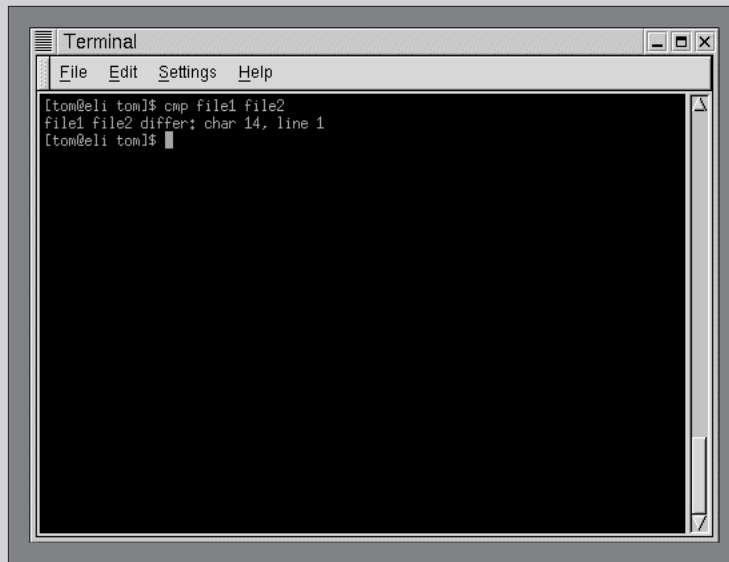


Figure 8-22: Cmp utility

Note: The `cmp` command displays nothing if the two files are identical.

Formatting Text in UNIX

Text formatting in UNIX involves preparing a text file with embedded typesetting commands and then processing the marked-up text file with a computer program. This program generates commands for the output device, such as a printer, a monitor, or some other typesetter. UNIX's `nroff` and `troff` commands are often used to process the embedded typesetting commands to format the output.

UNIX users have long used the `nroff` and `troff` commands to produce manuals, corporate reports, books, and newspapers. These programs evolved from an earlier program, `RUNOFF` (a utility created in the late 1970s), which read pure text with embedded codes to format and print a text-enriched report. An embedded code is a special sequence of characters that are included with the regular text in the file. The special codes are not printed, but interpreted as commands to perform text formatting operations. For example, there are codes to produce boldface print, center text, and underline certain lines.

Using embedded codes in text to produce enriched output provides the advantage of not needing additional word-processing programs to produce documents. You can use any editor that works with flat files, like `vi` or Notepad. In addition you can use

added features, such as hyperlinks to cross-reference other documents from within your document. You do need, however, an HTML browser program like Netscape, or UNIX utilities such as `nroff` and `troff`, to translate and execute the embedded hyperlink codes.

Linux introduced `groff`, which implements the features of both `nroff` and `troff`. Table 8-9 lists some embedded codes supported by `groff`.

Embedded Command	Meaning
<code>.ce n</code>	Center next <i>n</i> lines
<code>.ds C</code>	Center
<code>.ds R</code>	Right justify
<code>.p n</code>	Start a new paragraph indented <i>n</i> characters
<code>.sa 0</code>	Turn off justification
<code>.sa 1</code>	Turn on justification
<code>.ul n</code>	Underline the next <i>n</i> lines

Table 8-9: Some `groff` embedded commands

The `groff` command's usage follows this format:

Syntax

`groff [-Tdev]`

Dissection

The `-T` designates a device type, which specifies an output device such as ASCII to tell `groff` that the device is a typewriter-like device. In the format example, the device type (*dev*) is for the man pages. Some other device types include *ps* for postscript printers, *dvi* for TeX dvi format, and *lj4* for an HP LaserJet4-compatible printer.

Your task for Dominion Consulting is to produce a man page that contains the standard man page sections. You are to use `groff` to produce the man page. The format codes consist of tags and font-change commands that control the formatting, which you type into your man page document. The tags and font-change commands consist of:

- The `.TH` tag indicates the man page title, as well as the date and a version-number string. In the formatted man page, the version and date strings appear at the bottom of each page.

- The .SH tag indicates a section. (Section names usually appear in all upper-case characters on a man page.) Six common sections of a man page are:

NAME: the name of the command or program

SYNOPSIS: a brief description of the command or program

DESCRIPTION: a detailed description of the command or program

FILES: a list of files used by the command or program

SEE ALSO: a list of other commands or programs that are related to this one.

BUGS: a list of known bugs

- The .SS tag indicates the beginning of a subsection. For example, Options is a subsection of the DESCRIPTION section.
- The .TP tag indicates each item in the Options subsection.
- The \fB command changes the font to boldface, the \fI command changes the font to italic, the \fR command changes the font to roman, and the \fP command changes the font to its former setting.

You will now write a man page for the application, phmenu, that you completed in Chapter 7.

To write and format a man page:

- 1 Make sure you are in your ~/source directory. Recall that the ~ indicates your home directory.
- 2 Use the vi or Emacs editor to create the file phmenu.1. Type the following text into the file.

```
.TH PHMENU 1 "July 20, 2000" "phmenu Version 1.01"
.SH NAME
phmenu \- Menu for Dominion Employee Telephone Listings
.SH SYNOPSIS
\fB phmenu \fP
.SH DESCRIPTION
\fP Menu for maintaining employees' phones and job titles
\fP.
\fP Record includes phone number, name, dept, and date-hired
\fP.
.SS Options
.TP
\fB -v \fIView Phone List\fR
Display unformatted phone records.
.TP
\fB -p \fIPrint Phone List\fR
```

```

Corporate Phones report sorted by Employee Name.
.TP
\fb -a \fiAdd Phone to List\fr
Add new phone record.
.TP
\fb -s \fiSearch for Employee Phones\fr
Enter Name to search and retrieve phone record
.TP
\fb -d \fiDelete Phone\fr
Remove phone record
.SH FILES
.TP
\fc/home/dbadmin/source/corp_phones\fr

```

- 3 Save the file and exit the editor.
- 4 Test the man page by typing **groff -Tascii -man phmenu.1 | more** and then pressing **Enter**. Your screen should appear similar to Figure 8-23.

```

Terminal
File Edit Settings Help

[tom@eli tom]$ groff -Tascii -man phmenu.1 | more

PHMENU(1)                                PHMENU(1)

NAME
    phmenu - Menu for Dominion Employee Telephone Listings

SYNOPSIS
    phmenu

DESCRIPTION
    Menu for maintaining employees' phones and job titles.
    Record includes phone number, name, dept, and date-hired.

Options
    -v View Phone List
        Display unformatted phone records.

    -p Print Phone List
        Corporate Phones report sorted by Employee Name.

--More--

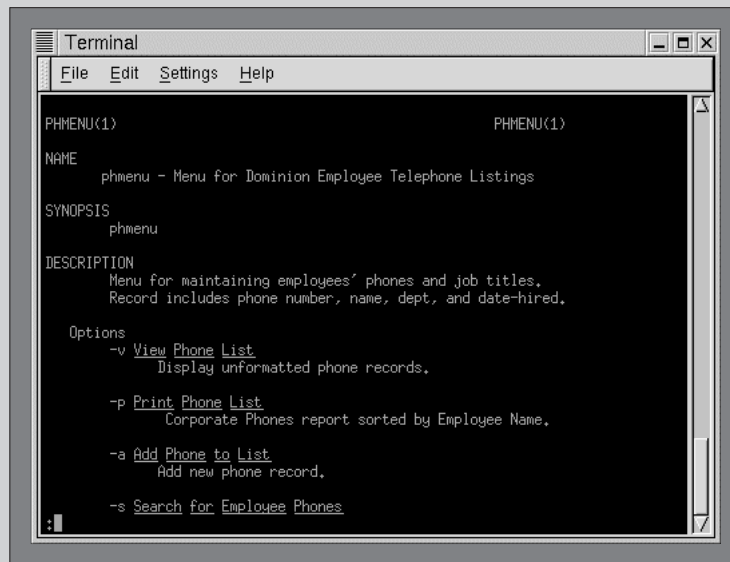
```

Figure 8-23: Phmenu man page

help

If you find any formatting discrepancies, check the dot commands and any embedded font changes against the code you typed in Step 2.

- 5 Press **q** to exit the more command, and then test your new man page by typing **man ./phmenu.1** and then pressing **Enter**. Your screen should be similar to Figure 8-24.



```
Terminal
File Edit Settings Help

PHMENU(1)                                PHMENU(1)

NAME
    phmenu - Menu for Dominion Employee Telephone Listings

SYNOPSIS
    phmenu

DESCRIPTION
    Menu for maintaining employees' phones and job titles.
    Record includes phone number, name, dept, and date-hired.

Options
    -v View Phone List
        Display unformatted phone records.

    -p Print Phone List
        Corporate Phones report sorted by Employee Name.

    -a Add Phone to List
        Add new phone record.

    -s Search for Employee Phones
```

Figure 8-24: Phmenu man page viewed with the man program

When you are satisfied with the man page format, you can make it available to others by copying it (while logged on as root) to one of the `/man` directories. All man pages are stored in subdirectories of the `/usr/man` directory. These subdirectories have names such as `man1`, `man2`, `man3`, and `man4`. All man pages in `man1` are identified with a common suffix, `.1`, so `phmenu.1` is copied to `/usr/man/man1`. The suffix number represents the version number of the man page.

When you request a man page using the `man` program, you specify the version number you want to see by placing the number after the name. (If you type only the name, `man` looks recursively for the page through all the subdirectories, starting with `/usr/man/man1`, and then displays the first match.) For example, if you want `man` to print the second version of the `break` command, follow the `man` command with `break 2`.

Note: You need superuser privileges to copy `phmenu.1` into the “root-owned” directory, `/usr/man/man1`. If you are not logged on to the root account, use the `su` command instead.

To copy the man page into a `/man` directory:

- 1 Type `su` and then press **Enter**.
- 2 Enter the root password and then press **Enter**.
- 3 Type `cp phmenu.1 /usr/man/man1` to copy the man page to the man directory, and then press **Enter**.

- 4 To exit from superuser mode, type **exit** and then press **Enter**.
- 5 Test that this file has been correctly copied by typing **man phmenu** and then pressing **Enter**.

In this lesson you learned to check the spelling of a document using the `ispell` utility. You learned to use the `cmp` command to check two files that seem similar, to determine any differences between them. Finally, you learned to create your own man page with the `groff` utility and test it with the `man` program.



S U M M A R Y

- The spell-checking utility, `ispell`, scans a text document for typing errors.
- Text formatting in UNIX involves preparing a text file with embedded typesetting commands and then processing the marked-up text file with a computer program that generates commands for the output device.
- The text containing the embedded typesetting commands is processed (read) by a program like UNIX's `nroff` and `troff` utility programs that formats the output.
- Linux introduced `groff`, which implements the features of both `nroff` and `troff`.
- Those who have superuser (root) privileges, such as the system administrator, most often create man pages.



R E V I E W Q U E S T I O N S

1. The `groff` command is a(n) _____.
 - a. text formatting utility
 - b. man page utility
 - c. search text utility
 - d. editor
2. True or false: The `ispell` utility only displays misspelled words. It does not allow you to change the misspelled words.
3. The embedded commands used by `groff` to format a text file are placed in the _____.
 - a. command line
 - b. file itself
 - c. same directory as the text file, in a special command file
 - d. `man1` directory

4. To produce the fourth version of the tty man page, type _____.
 - a. `man tty 4`
 - b. `man 4 tty`
 - c. `man tty.4`
 - d. `man tty`
5. Which of these commands gives an ordinary user the ability to print a man page from the current directory?
 - a. `man ./phmenu.1`
 - b. `man phmenu`
 - c. `man menu 1`
 - d. `man -1 phmenu`
6. The man page format codes consist of _____ and _____ commands.
 - a. tag, font change
 - b. dot, dot dot
 - c. header, footer line
 - d. tag, sub header
7. The title of a man page is denoted by the _____ embedded command.
 - a. `.T`
 - b. `.TITLE`
 - c. `.NAME`
 - d. `.TH`
8. The `\fB` embedded command causes _____.
 - a. the font to change to bold
 - b. the text to appear in brackets
 - c. the line to appear at the bottom of the screen
 - d. the font to change back to the previous font
9. The man pages are normally copied into a subdirectory of _____ to make them available to all users.
 - a. `/etc`
 - b. `/home`
 - c. `/usr`
 - d. `/usr/man`
10. Which of these commands tests your man page?
 - a. `groff -T ascii phmenu.1 | more`
 - b. `groff -T ascii -man phmenu.1 | more`
 - c. `man -T ascii phmenu.1`
 - d. `troff -T ascii phmenu.1`

EXERCISES

1. Use vi to create the document Spellcheck, and then type the following four lines, with spelling errors:

```
Rosus are red,
Vilets are blu,
I luv Linux,
and So should Yu.
```

2. Make a copy of the file called Spellcheck.bak. Run ispell and correct the original document, Spellcheck.
3. After you correct the Spellcheck file (created in Exercise 1), compare it to the Spellcheck.bak file with the cmp command.
4. Create a simple man page to describe a fictitious program, Movies, that searches a movie-rental database for movies using optional search-argument codes. The search-argument codes are S (enter star's name), M (enter movie name), and D (enter director's name).

.....
Remember to enter a suffix (.1) to the man page document name and when you run man. As an ordinary user, you must enter the name plus the suffix.

5. Run the Movies man page from your current directory by typing **\$ groff -T ascii ./phmenu.1 | more** and then pressing **Enter**. What happened? Explain.
6. Now run the movies man page from your current directory by typing **\$ groff -T ascii -man ./phmenu.1** and then pressing **Enter**. What happened? Explain. How would you fix the problem?

DISCOVERY EXERCISES

1. Use the ispell utility on the Phmenu.1 man page you created in this chapter. Why must you be careful not to change all misspelled words the utility finds?
2. Edit the Phmenu.1 file and add a new section named SEE ALSO. Under this section, list the files:


```
Phoneadd
Phlist1
```
3. Save and test the revised file using the groff and man programs.
4. Edit the Phmenu.1 file and add a new section named BUGS. Under this section, list a line that reads:


```
None Known
```
5. Save and test the revised file using the groff and man programs.
6. Edit the Phmenu.1 file and add a new section named AUTHOR. Under this section, list your name. Save and test the revised file using the groff and man programs.